Integrating Open Hypermedia Systems with the World Wide Web

Kenneth M. Anderson
Department of Information and Computer Science
University of California, Irvine
Irvine, CA 92697-3425
FAX +1.714-824-4056
Email: kanderso@ics.uci.edu

ABSTRACT
Research on open hypermedia systems (OHSs) has been conducted since the late Eighties [10]. These systems employ a variety of techniques to provide hypermedia services to a diverse range of applications. The World Wide Web is the largest distributed hypermedia system in use and was developed largely independent of the research in OHSs. The popularity of the Web along with problems inherent in its design has motivated OHS researchers to integrate their systems with it. This research has primarily focused on enhancing the functionality of the Web via the services of an OHS. This paper presents three experiments exploring the integration of the Chimera OHS with the Web. While one of the experiments indeed describes work which enhances the Web, the other two investigate ways in which the Web can beneficially enhance an OHS. The paper concludes with a call for both communities to continue research which focuses on integration.

KEYWORDS: Chimera, integration, open hypermedia systems, World Wide Web

1 INTRODUCTION

Open hypermedia systems (OHSs) provide hypermedia services to client applications via an open distributed architecture [15, 12]. An OHS's primary responsibility is the management of hypermedia links for its client applications. The OHS ensures link consistency in the face of change. It maintains this consistency by constructing a model of the hypermedia structures created by its client applications. For instance, if a client application deletes an anchor, the OHS responds by removing the anchor from all links which contained it. This model of the hyperweb enables quick traversal and search over the hypermedia structures. This, in turn, enables various analyses and visualizations to be performed on the hyperweb.[1]

Unlike monolithic hypermedia systems (such as KMS [1]), an OHS can integrate a wide variety of clients through a suite of integration techniques [5]. In addition, the architecture of an OHS is distributed in terms of execution, data, and time. The first category refers to the concurrent execution of both clients and servers across a set of host machines. The second category implies that clients and servers can store and retrieve their persistent information either locally or on a remote file system or object store. Distribution across time refers to the hypermedia system's support for collaboration between its end-users.

Most open hypermedia systems handle some aspects of distribution for each of the three areas. However the degree of support for each area differs between systems, and the interaction with these features or the protocols employed are non-standard and vary across systems. The result is that a typical end-user can not work with multiple open hypermedia systems easily (few, if any, interoperate[2]), and the experience gained in learning one system can not be employed in using the next (assuming the systems are freely available).[3]

The World Wide Web (WWW) [3] is the largest distributed hypermedia system in use. It provides support for each of the three areas described above and makes use of data formats (e.g. HTML) and access protocols (e.g. HTTP) which are open, extensible, and standard. The existence of these standards has enabled the WWW to be pervasively available across platforms and its low entry barrier to use has encouraged widespread adoption among end-users.

Despite its popularity and support for global distribution, the WWW has several weaknesses when examined from within

Hypertext 97, Southampton UK

---

1. This material is based upon work sponsored by the Air Force Materiel Command and the Advanced Research Projects Agency under Contract Number F30602-94-C-0218. The content of the information does not necessarily reflect the position or the policy of the Government and no official endorsement should be inferred.
2. The OHS community is currently addressing the issues of interoperability (e.g. <http://www.csdl.tamu.edu/ohs/>).
3. Few systems are. Notable exceptions are Chimera [2], DHM [6], the Distributed Link Service [4], and Hyper-G [9]. Both Microcosm [8] and Hyper-G are also now available as commercial products.

the rich context of hypermedia research. First, the hypermedia model of HTML, *embedded links*, lacks separation between the data being linked and the links themselves. This leads to a variety of problems including link maintenance in the face of change and dangling links. In addition, only point-to-point links are supported making it impossible to handle *n*-ary, or multi-headed, links. Second, standard hypermedia features such as guided tours and overviews are not provided automatically, and the user must either generate these features by hand or depend upon an HTML-generator to include them. Finally, annotation is prohibited by the standard protocols. A limited form of annotation can be provided through the use of CGI scripts. Unfortunately, these scripts are difficult to develop and the resulting mechanism is non-standard.

A recent trend in open hypermedia research is the integration of OHSs and the Web.[4] The motivation behind this research is the goal of producing a hybrid system where the advantages provided by one, help to counteract the disadvantages of the other. OHSs can be used to provide sophisticated hypermedia features not present in the Web and enable link consistency and management functionality over data stored in the Web [8]. In turn, the Web can provide greater distribution for OHS information, and standards to increase the interoperability and ease-of-use of OHSs. This paper reports on three experiments conducted in this area between the Chimera OHS [2] and the WWW. This work provides a practical demonstration of the benefits achieved by this type of integration and serves as a catalyst for future work.

This paper is organized as follows. Related work is considered within the context of a simple taxonomy. Then, three experiments and their demonstrated benefits are discussed. Future work is described next, after which the paper concludes with a discussion about the lessons this work has for the Web and OHS communities.

2 RELATED WORK

A simple taxonomy of possible integrations between OHSs and the Web provides context for a discussion of related work. The taxonomy is constructed by considering the intersections between the four major architectural elements common to both systems: clients, servers, protocols, and data formats. All possible intersections will not be considered, instead only areas in which work has occurred are discussed. Hybrid integrations drawing on more than one intersection are also described.

4. This research is timely in that solutions to problems which OHSs can solve are starting to come from the Web community without the use of an existing OHS. See for instance [11]. OHS researchers have the advantage of leveraging existing systems to solve these problems, rather than building a new system from scratch.

2.1 OHS Data to WWW Data

One simple integration is to produce tools which translate links and content within an OHS into HTML. While most likely there will be restrictions on the type of information and functionality that can be translated, this integration has the important property of leaving the elements of both systems unmodified. This eases the amount of work required to achieve the integration. The only effort, then, is the development of the translators which is significantly less effort than modifying the server of an OHS or extending a WWW server. This type of integration allows an end-user to enjoy the benefits of creating content within an OHS while at the same time retaining the ability to then publish the information on the Web.

The Microcosm group [8] has done work in this area by producing a tool which converts Microcosm *applications* into a set of linked HTML documents. This conversion has a number of restrictions: only RTF documents and BMP images can be translated by the tool into HTML and GIF respectively. In terms of hypermedia functionality, the tool can easily convert Microcosm *specific* links and buttons into HTML links. *Local* and *generic* links can be converted as well, however this involves searching the text of every document included in the application to find all possible destinations. [8] also points out the difficulty of translating links with multiple destinations into HTML and discusses several strategies to deal with this problem.

2.2 WWW Client as OHS Client

Another simple integration is integrating a WWW client (i.e. a Web browser) with an OHS. Such an integration is reported in [8] for Microcosm and in [2] for Chimera. The former integration was performed using Microcosm's *Universal Viewer* and the latter was performed using a Chimera *wrapper*. This type of integration achieves a level of functionality such that HTML pages can be linked "out of the Web" and into the other clients supported by the OHS. This has the advantage, for instance, of allowing developers to write their documentation in HTML and then link it directly to the relevant application.

A more advanced version of this type of integration is illustrated in the work to integrate DHM with the WWW [7]. That paper discusses three approaches to integrating WWW clients with DHM using mechanisms such as applets, Javascript, and browser plug-ins. It concludes that the technology is not yet sufficient for a platform/browser-independent solution. However the integrations achieved do show promise for the technique of combining several Web mechanisms in tandem to achieve a strong integration.

When targeting a specific browser or platform, the effort to perform this type of integration is low. Microcosm's Universal Viewer makes the process trivial under Microsoft Windows, and creating a Chimera wrapper for an application with an external application program interface (API) is not difficult. The disadvantage to this approach is that current Web mechanisms are not set up to help outside systems override the default hypermedia services of the WWW. Instead these mechanisms are tuned to help display new

media types within a browser or to implement simple client-server applications. The feedback gained from the difficulties encountered by work such as the DHM effort should help the Web community respond with improved mechanisms to achieve more universal integrations.

2.3 WWW Server as OHS Client

A deeper integration can occur by extending a WWW Server to be a client of an OHS. In this approach, users gain the ability to access the functionality of the OHS from a standard unmodified Web browser. The Web server is modified to make calls on the OHS in response to requests for certain URLs. Based on the response to these requests, the server generates HTML to be sent back to the client which presumably contains markup that manifests the presence of the OHS.

This approach has the principle benefit of separating links from Web documents, since the links are generated and inserted dynamically. In addition, authors can continue to create hyperwebs in their favorite OHS and have the results transparently available via the Web.

Implementation mechanisms for this integration include extending the WWW server directly via changes to its source code, using a server plug-in, or implementing a CGI script. The first two options provide performance benefits, while the third is stronger with respect to portability between Web servers. One other implementation mechanism for this approach is to use a client-side HTTP proxy. This mechanism has the advantage of allowing the user to specify when they wish to access the OHS's services, and the additional benefit that the work required to modify the Web server is avoided. The disadvantage of this approach is that the developers of the proxy must deal with cross-platform portability issues.

The Microcosm group has performed work in this area in a project called the Distributed Link Service (DLS) [4, 8]. In this system, Web servers have been extended to access distributed linkbases and filters. In response to client requests, information in the linkbases is "compiled" into the HTML document as URLs on the fly.

The work integrating DHM with the WWW [7] also falls under this category. The client integrations mentioned in Section 2.2 access URLs which invoke cgi-scripts which in turn access the DHM server for external anchor and link information.

Our work in this area is described in Section 3.1.

2.4 Hybrid Integration

Hybrid integrations apply two or more of the techniques described in this taxonomy and apply them simultaneously in one hypermedia application. A popular hybrid integration is to combine the strategies of the previous two sections. The result is a powerful hypermedia environment for the end-user. In addition to using normal OHS clients, they can now effortlessly link in and out of Web documents. In addition, due to the server integration, anchors and links created on Web documents are stored externally in the OHS's hypermedia database yet appear as standard Web links when rendered by a Web browser. A user also has the power to configure and access other services of the OHS from within the Web browser due to the client integration.

The DHM work [7], described previously, is an example of this type of hybrid. In addition, DLS [4] achieves this hybrid integration via a modification to the Microcosm Universal Viewer which allows users to access standard Microcosm features through the integrated Web server described in Section 2.3.

Other hybrid integrations are discussed in Section 3.2 and Section 3.3.

2.5 OHS Server as WWW Server

One interesting form of integration is to modify an OHS server to masquerade as a WWW server. While the results of this integration from the standpoint of an end-user is similar to the previous section, it offers greater flexibility to the OHS. Performance is improved since the OHS receives HTTP requests directly and has immediate access to both the requested document and its hypermedia database. Additional services can be made available since more of the OHS's API can be exposed to WWW clients. However this integration involves significant developmental effort since implementing the functionality of a Web server is not trivial. This complexity can be mitigated by limiting the scope of URLs handled by the OHS server.

The most notable work in this area has been performed by the Graz University of Technology, Austria, through its Hyper-G project [9]. This system is an advanced hypermedia authoring environment which integrates with the Web by having its server masquerade as a WWW server. Normally Hyper-G information is best viewed with a proprietary client called Harmony, however Hyper-G servers have the ability to receive requests from standard Web clients and translate their information (with an associated loss of functionality) into HTML. Users can thus gain access to the large amount of structural information stored in Hyper-G from any standard WWW client.

2.6 WWW Protocols Within an OHS

This approach to integration employs WWW protocols within an OHS to help leverage their distribution and standardization benefits. For instance, the hierarchical structure employed by WWW servers to organize information within websites can be used to organize the management of hyperwebs within an OHS. In addition, the use of URLs to locate OHS components allows for these components to be globally distributed, yet accessible in a way familiar to all Web users.

HyperDisco [13] has demonstrated the ability to access *workspaces* distributed across the Internet [14]. These workspaces contain hypermedia information as well as content. The HyperDisco *tool integrator* has been extended with a workspace class which enables it to communicate and manipulate with any workspace regardless of its location.

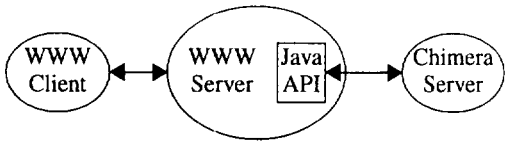


Figure 1: Architecture for Experiment 1. A standard WWW server makes use of the Java API of Chimera 1.0 to communicate with the Chimera server. It interprets HTTP requests as operations on the Chimera server and generates HTML to display the results.

Web protocols were not utilized to enable tool integrators to discover remote workspaces, however. Instead, a name service is used which places a small amount of work on the user in terms of maintenance (users must manually enter connection information for remote workspaces) but offers more transparency and flexibility than the URL protocol (the user can assign names to workspaces, and allow the name service to map these names to the proper workspace).

The experiment described in Section 3.2 explores this integration approach in detail, and compares it with the HyperDisco approach.

3 INTEGRATION EXPERIMENTS

Chimera is an open hypermedia system with an emphasis on the flexible modeling of software engineering environments [2]. It consists of a server providing hypermedia services to client applications with full link consistency and management capabilities. The first integration experiment utilizes the initial version of Chimera to demonstrate a twist on integrating a WWW server with an OHS. The next experiment describes the development of a new version of Chimera which utilizes standard Web protocols. This allows it to locate and connect its various components and simplify the management of Chimera hyperwebs. The third experiment describes the use of a hybrid OHS-WWW integration to provide the presence of Chimera throughout the Web.

3.1 Extending a WWW Server

In this experiment, which falls under the domain of Section 2.3 (WWW Server as OHS Client), a WWW server written in Java was extended to make use of the Java API of Chimera. This extension enabled the server to provide access to the anchors and links stored in a Chimera hyperweb (see Figure 1). The Java server connects to Chimera at start-up and interprets URLs as requests on the Chimera server. A request to its root URL causes the server to retrieve all the anchors from the current hyperweb, displaying them as a list of anchor identifiers. Clicking on one of the listed anchors causes the server to retrieve additional information such as its associated *view*, including both the *viewer* and *object* components.[5] In particular, if the view's object is a text or GIF file, this file is retrieved and displayed. Next, all links associated with the current anchor appear. This list displays the set of anchors contained in each link. Clicking on any of the destinations initiates a link traversal and retrieves the associated information. Users can also directly access a link or anchor by specifying its identifier with a URL of the form <http://www.some.domain/[anchor | link]/###>.

In this fashion, users are able to traverse a Chimera hyperweb without having to invoke the associated clients which normally handle the display and management of anchors and view information. This lightweight browsing mechanism enables the state of a hyperweb to be quickly assessed by a user.

In the future, this server can be extended to display anchors for the data formats that it understands more naturally, thereby becoming a proxy viewer for that datatype. In addition, a simple forms interface would allow a user to directly query the Chimera server, in a way standard throughout the Web. This work is similar to the work performed by DLS [4] in extending a WWW server to access DLS filters when retrieving HTML documents. The difference here is that our work simply utilizes existing Web mechanisms to display the hypermedia structures contained in a Chimera hyperweb and does not attempt to dynamically modify the contents of an HTML document based on the current hyperweb.

3.2 Utilizing Web Protocols Within an OHS

The initial version of Chimera contained some implementation restrictions which hindered the distributed aspects of the system. In addition, it had a number of user interface quirks which prevented new users from learning the system quickly. This experiment simultaneously relaxes these restrictions and lessens the learning curve for new users by utilizing Web protocols and standard Web interaction styles. The problems encountered in the initial version of Chimera are now discussed, and then one solution developed during the course of the experiment is presented.

In order for a user to access a hyperweb in Chimera, the user's account, the user's Chimera clients, the Chimera server, and the desired hyperweb all had to reside on the same network file system. This restriction limited Chimera's support for distribution to a LAN. The reason for this restriction involved the ability to access various text files which contained the location of the hyperweb and the connection information for the Chimera server.

With respect to user-interface issues, Chimera hyperwebs were difficult to manipulate and use effectively. The only way to reference a hyperweb was with its absolute path name. Links between hyperwebs were not supported, forcing users to place all information into one hyperweb rather than distributing information across related hyperwebs. Finally, Chimera provided no mechanism to discover the existence of hyperwebs created by other users. All of these issues are addressed via integration with the Web.

The new version of Chimera has an architecture split between a user's environment and external websites (see Figure 2). A user's space contains Chimera clients, a Web

5. See [2] for more information about Chimera's hypermedia concepts.

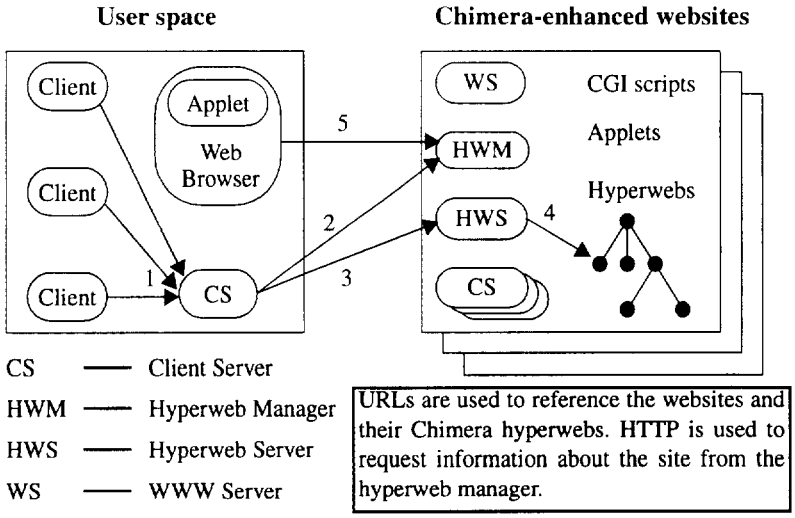


Figure 2: Interactions within the Chimera 2.0 architecture. This diagram shows some of the interactions possible between Chimera 2.0 components: (1) A user can invoke Chimera clients which interact with the local client server. (2) The local client server presents a user-interface which allows the URL of a Chimera website to be specified. The client server connects to the hyperweb manager of the site to determine the connection information of the site's hyperweb server. (3) It can then connect to the hyperweb server and allow the user to select among the available hyperwebs. (4) Once selected, the hyperweb server loads the hyperweb into memory allowing it to be manipulated by the active clients. (5) The hyperweb manager, which has a pre-defined port number, can handle direct HTTP requests for information about the site. This architecture, developed for the second experiment, is discussed in detail in Section 3.2. The use of applets is discussed in detail in Figure 4.

browser, and a client server. A Chimera website contains a WWW server, a hyperweb manager, a hyperweb server, and a set of client servers.

The original Chimera server's duties have been split between the client server and hyperweb server. The client server provides the user with an interface to specify the Chimera website of interest and to select among the hyperwebs stored there (both via URLs). It also manages the connection to the site's hyperweb manager and hyperweb server, and routes to the hyperweb server the requests of the user's clients to manipulate the current hyperweb. This is similar to HyperDisco's enhanced tool integrator described in [14]. The main difference is that the client server does not utilize a name service to locate the remote servers. Instead, URLs are used to initiate HTTP requests to establish the connection between the client server and the remote site. Once connected, it switches to Chimera's native protocol for more efficient communication.

The hyperweb server is responsible for the persistent storage and management of a Chimera website's hyperwebs. Each hyperweb contains instances of Chimera's hypermedia concepts and is located under the WWW server's namespace. This allows them to be easily referenced by URL (for instance, <http://www.some.domain/chimera/web1/>) and include hierarchical sub-webs. These hyperwebs are similar to the workspaces described in [14] only insofar as they contain hypermedia information and can be located anywhere on the Internet. HyperDisco workspaces have the additional ability to store documents and other application-specific data. These documents can be retrieved remotely by the HyperDisco tool integrator. Since Chimera maintains a strict separation between application-specific content and hypermedia information [2], it does not attempt to provide these features.

The Web browser in a user's environment can download Chimera clients in the form of Java applets. Since applet communication is restricted to its server host, these applets connect to the client servers running on the Chimera website. They receive the necessary connection information from the site's hyperweb manager (discussed next). Once connected they can use the client server to connect to any Chimera website not just the one from which they were downloaded.

The hyperweb manager enables the creation and deletion of hyperwebs. In addition, it tracks the state of its website including the connection information for the other Chimera-related servers. The hyperweb manager is able to service HTTP requests for this and other information directly, enabling remote clients to connect to these servers.[6] For instance, <http://www.some.domain:hwm/servers> queries the hyperweb manager for the connection information about the client servers in its domain.

A URL of the form <http://www.some.domain:hwm/chimera/hyperwebs> is used to determine the names of the hyperwebs available for a site. The URL <http://www.some.domain:hwm/chimera/hyperwebserver> retrieves the connection information for the site's hyperweb server. This ability of the hyperweb

6. A potential bottleneck here is mitigated by servicing multiple HTTP requests in parallel, each in a separate thread.

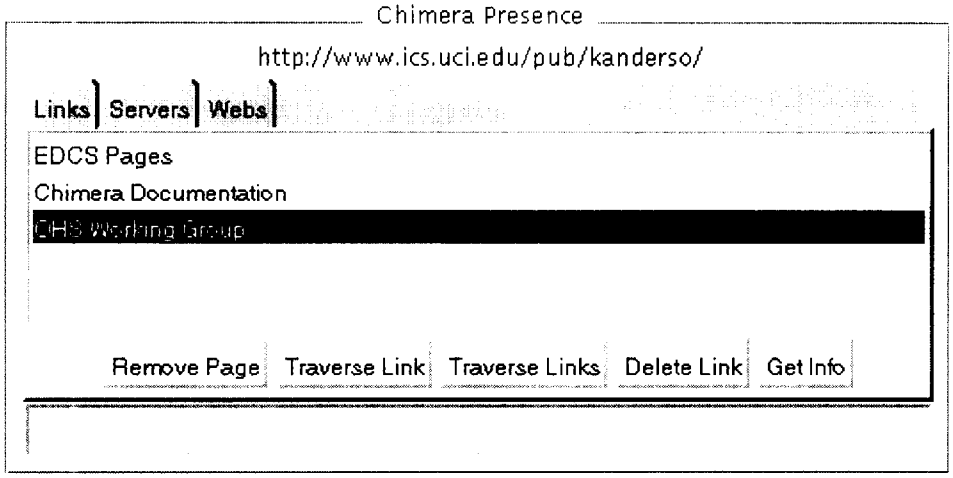


Figure 3: The Chimera Presence applet (third experiment). The applet presents a user-interface divided into panes. The Servers pane allows users to specify the desired Chimera website while the Webs pane allows them to select among the site's hyperwebs. The Links pane (shown above) displays all of the links contained in the hyperweb and provides operations to manipulate links, initiate traversals, examine a link in detail, and add or remove the Web page to or from the selected link.

manager to receive HTTP requests directly is similar to the gateways employed by Hyper-G [9] to allow standard Web clients access to its hypermedia information. The primary difference is that the hyperweb manager does not yet fully implement the responsibilities of a full WWW server. Instead it only responds to a small set of URLs and ignores all other requests.

This new version of Chimera is still under development yet it has already addressed some of the major problems with the previous version. Internet distribution is achieved through the use of URLs. These URLs contact the hyperweb manager to retrieve a server's connection information, which used to be specified in text files on a shared file system.

In addition, the ability to discover new hyperwebs is enabled via the *hyperwebs* URL which returns information on all hyperwebs available at a particular site. The problem now shifts to discovering Chimera-enhanced websites. However this problem can easily be solved with existing Web mechanisms, such as a query on an Internet search engine or a Web page at the Chimera website which tracks all known sites. The former is possible since the root URL of a Chimera site (i.e. <http://www.some.domain/chimera/> is configured to return a page containing keywords indicating a Chimera-enhanced site.

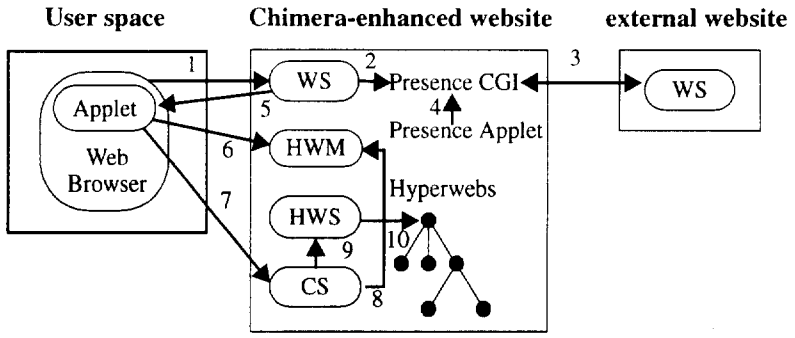
Finally, hyperwebs are easier to manage and understand due to their now explicit hierarchical structure and the interface provided by the hyperweb manager which provides operations such as create, delete, rename, and get info. Links between hyperwebs are supported as well. In the original version of Chimera, the Chimera server could only manage one hyperweb at a time. This made it impossible for it to create links which spanned hyperwebs. Now with the new separation between the client and hyperweb servers, a client can have a client server connect to multiple hyperweb servers and begin the construction of a new link. Each anchor in the link is attributed with the URL of its hyperweb. A copy of this link is then placed in each hyperweb. At a later time, the client server can start a link traversal over the link from any hyperweb and use the URLs associated with each anchor to traverse to the correct hyperweb. This solution is similar to the solution used by HyperDisco as described in [14]. The primary difference being the use of a URL to indicate the remote hyperwebs in place of a workspace name.

Finally, the new version of Chimera is an example of a hybrid integration with aspects falling under the domains of Section 2.5 (OHS Server as WWW Server), since the hyperweb manager masquerades as a WWW server, and Section 2.6 (WWW Protocols Within an OHS) since WWW protocols are used throughout the architecture.

3.3 Providing a Pervasive OHS Presence on the Web

The third experiment builds on the work of the previous section to provide Chimera with a pervasive presence on the Web. The idea is to attach a Java applet which provides access to Chimera's services to every Web page a user visits. The user can create and manipulate links, switch between hyperwebs, and initiate link traversals all from within the applet's user-interface (see Figure 3). Since the Java virtual machine has been ported to a variety of platforms and Web browsers, this Web mechanism provides Chimera services cross-platform for the first time.

Unfortunately Java's applet mechanism is unable to achieve the goal of pervasive presence due to the security restrictions placed on applets. First, an applet is not allowed to get the contents of a URL because URL content handlers fall outside the set of legal classes for applets. This prevents the applet from directly utilizing the URL mechanism described above to contact the Chimera hyperweb manager. Second, an



The legend for this diagram is the same as in Figure 2. Step 8 demonstrates the client server connecting to the local website's hyperweb manager. This was done to simplify the diagram. In reality, the applet can direct the client server to connect to any Chimera-enhanced website since it passes a URL to indicate the desired site.

Figure 4: The architecture of the third experiment. The process outlined above is as follows: (1) The user fills out an HTML form specifying their user name and target destination. (2) The web server receives this form and invokes its associated cgi-script. (3) The cgi-script retrieves the target document, modifies its anchors (see Figure 5). and (4) inserts the applet tag at the end of the document. (5) As a result the applet is downloaded into the web browser. (6) After initialization, the applet contacts (via an HTTP request) the hyperweb manager located on its server site. (7) The hyperweb manager provides it with the contact information for a client server which it then contacts. (8) After determining the target site (either built in to the applet or provided by the user), the client server contacts the target's hyperweb manager. (9) This enables the user to select a hyperweb, which is accessed via the site's hyperweb server. (10) If the hyperweb is not currently active, the hyperweb server loads it into memory. When traversing to a new page, the cgi-script is contacted once again and the process repeats. However, this time the applet defaults to using the website and hyperweb selected on the previous iteration.

applet is only allowed to establish a socket connection to the machine from which it was downloaded. This means that the applet can only be attached to pages which come from a Chimera-enhanced website. Third, an applet is not able to communicate state between instances of itself located on separate Web pages. Among other things, this implies that each time the user traverses to another page, they will have to re-specify information such as their user name and the current website and hyperweb.

The solution for this experiment involved three parts. First, the applet implements the HTTP request on the hyperweb manager directly. Second, a CGI script is used to attach the applet, called Chimera Presence to a Web page. Third, an HTML form is used to initialize the CGI script with the user's name. In turn, the CGI script sets an HTTP cookie to save the name of the user for each subsequent applet invocation. Each part of the solution is now described and the architecture of the experiment is shown in Figure 4.

The problem of an applet not being able to access the contents of URLs is solved by implementing the HTTP request on the hyperweb manager directly. The hyperweb manager can be contacted since it lives on the Chimera-enhanced website from which the applet was downloaded. The hyperweb manager makes use of a pre-defined port number avoiding the need for a name service in order for clients to locate it.

The CGI script applies a technique similar to the service provided by <http://www.anonymizer.com/>. The CGI script is given one parameter: the URL of a Web page. The CGI retrieves the contents of the specified URL, modifies the links within to be prefixed by the URL of the CGI script, and appends the HTLM necessary to include the Chimera Presence applet at the end of the page (see Figure 5).

The CGI script solves the latter two problems mentioned above in three steps. First, the CGI script modifies Web pages to always call the CGI script for subsequent link traversals. This allows the CGI script to attach the applet to each page the user visits via direct link traversal. In this manner, the applet appears on each page providing pervasive access to Chimera. The applet will not appear in the case where a user enters a URL directly into the Web browser. One solution to this problem is through the use of an HTML form attached beneath the applet. Regardless, the ability to remove the presence of the OHS is needed so that the user has the power to choose when to access these services.

Second, the script executes on a Chimera-enhanced website which means the applet will be able to communicate with the hyperweb manager and client server on that site. Third, the CGI script sets applet parameters to specify the current page and the user's name. It receives the user's name when first invoked by the HTML form, and by the HTTP cookie that it sets on each subsequent invocation. As the current site and hyperweb are changed by the user from within the applet, the applet stores this information with the hyperweb manager indexed by user name. When invoked subsequently, the applet retrieves the current values for the site and hyperweb from the hyperweb manager using the user name supplied by the CGI script. This baroque process is necessitated by the fact that the Java security manager blocks an applet from determining the name of its user. The proposed trusted applet

```
Before:

<HTML><HEAD><TITLE>Example One</TITLE></HEAD>
<BODY>Please visit the <A HREF="http://www.ics.uci.edu/pub/chimera/">Chimera web page</A>.
</BODY></HTML>

After:

<HTML><HEAD><TITLE>Example One</TITLE></HEAD>
<BODY>Please visit the <A HREF= "http://www.some.domain/chimera/chimeraPresence?http://
www.ics.uci.edu/pub/chimera/">
Chimera web page</A>.
<applet CODE="presence.class" CODEBASE="/applets/presence" WIDTH=500 HEIGHT=250>
<param name="page" value="http://www.ics.uci.edu/pub/chimera/">
<param name="user" value="kanderso">
The Chimera Presence Applet</applet>
</BODY></HTML>
```

Figure 5: Output of the Chimera Presence CGI script. This figure demonstrates the output of the CGI script associated with the Chimera Presence applet. The CGI script inserts its URL as a prefix to any URLs contained in the target HTML document, and inserts the applet at the end of the document.

mechanism of Java may help to eliminate this part of the solution in the future.

This experiment is another example of a hybrid integration. The applet's functionality falls within the category discussed in Section 2.2 (WWW Client as OHS Client), Chimera 2.0 is a hybrid integration in and of itself as discussed in Section 3.2, and the CGI script enables the server to provide OHS functionality, thus falling within the domain discussed in Section 2.3.

4 FUTURE WORK

These experiments have demonstrated a few of the benefits gained via the integration of an OHS with the WWW. They have also made explicit, areas for improvement and new limitations which must be addressed. One important issue is performance, especially in terms of network communication. Currently socket communication in Java is not as fast as it would be in a compiled language. Speed improvements have been promised in future versions of Java.

The hyperweb manager's ability to handle HTTP requests has turned out to be useful in enabling the global distribution achieved by the new version of Chimera, as well as providing ways to monitor and configure the system from within a standard WWW client. In fact, plans are in place to enable all Chimera 2.0 servers with this ability. This enables the exploration of the benefits gained by making the APIs of these servers accessible via HTTP.

The new version of Chimera is still under development and one area of exploration is its new support for cooperative work enabled by its integration with the Web. The URL mechanism of connecting to client servers increases the range and number of users that can connect to a Chimera hyperweb. The full range of asynchronous and synchronous collaborative activities that can ensue from this increased distribution will be explored. Chimera already has an advanced notification system for distributing the events occurring within a hyperweb to all connected clients. This system must now be evolved into a full-fledged awareness mechanism to inform users of the actions occurring within all connected hyperwebs. Algorithms useful for distributing these events through all relevant hyperweb and client servers will be investigated.

The Chimera Presence applet also suffers from performance issues in addition to the security restrictions discussed in Section 3.3. A few of the performance concerns will be addressed by features in future releases of Java. For instance, the ability to include all of the applet's classes in one compressed file will significantly decrease the amount of time it takes to load the applet into a Web browser.

The use of a client-side HTTP proxy will also be explored in order to replace the CGI script described in Section 3.3. This proxy would alleviate the time-consuming and error-prone process of parsing the retrieved page looking for links to modify. The primary concern with the use of the proxy is that applets can only contact the machine from which they were downloaded. If the client-side proxy cannot get the Web browser to download the applet from the correct site, it will be unable to communicate with the relevant Chimera servers.

A deeper issue is the design of presenting the applet to the user. The current choice of appending the applet to the bottom of the page was chosen for its simplicity. However there are obvious problems with this design such as not being able to see the applet in the case of long HTML pages. Given that users don't like to scroll to find information [9], it would be desirable to have the applet viewable at all times. This points to placing the applet in an HTML frame, but this leads to various compatibility problems. A satisfactory solution to this issue has not yet been achieved. Guidance for subsequent designs will be sought by conducting usability studies and examining techniques reported by other researchers (such as the techniques employed in [7]). As discussed in Section 2.2, the WWW community must respond to the difficulties encountered by hypermedia

researchers and provide better mechanisms for integrating advanced hypermedia features into the WWW.

5 CONCLUSIONS

Previous work in OHS-WWW integration has already demonstrated the benefits open hypermedia systems can provide to the WWW [4, 9]. In particular, the deficiencies of HTML's hypermedia data model can be avoided since anchors and links can be stored separately from content by the OHS and compiled into HTML dynamically upon request. The three experiments described here have illuminated the benefits gained by taking the integration the other direction that is, integrating the WWW into an OHS. The obvious benefits are the global distribution provided by WWW protocols and the re-use of standard Web user-interface mechanisms. The former allows servers, clients, and data to be located anywhere on the Internet, while the latter allows an OHS to leverage a user's experience with the Web by presenting a similar user-interface. A more important benefit is the deeper integration of hypermedia services into a user's environment achieved when these augmented OHSs are used as a basis for delivering OHS functionality to the Web.

The third experiment has lessons for both the hypermedia and Web communities. In particular, it stretches the limits of the intended design of URLs, Java applets, and cgi-scripts by producing a hypermedia application which depends on their combined use, and flawless interaction with an underlying OHS. It demonstrates to the hypermedia community that the Web is a viable platform for experimentation whose reusable mechanisms are cross-platform and accessible to most end-users. The WWW community benefits both from seeing the increased power which can be brought to the Web as well as receiving feedback on their existing mechanisms. While the third experiment succeeded in achieving its goals, there are problems with the final solution (discussed in the previous section) which prevent it from being "industrial-strength." It is thus important for both communities to enter a feedback loop and continue to improve the mechanisms which enable their integration.

ACKNOWLEDGMENTS

Peyman Oreizy performed the work of the first experiment described in this paper. Thanks also goes to Richard N. Taylor for his constant guidance throughout the Chimera project. Uffe K. Wiil just finished a brief visit here at UCI. I greatly enjoyed the resulting interchange of ideas and his comments on my research. The conference reviewers provided excellent suggestions, and Randy Trigg offered kind assistance in fine-tuning the paper. Finally, the OHS research community, especially the Microcosm, DHM, and HyperDisco projects, offers a vibrant area in which to perform research, consistently producing high-quality results. The resulting challenge in producing work which meets these high standards is a constant source of motivation.

REFERENCES

1. Akscyn, R. M., McCracken, D. L., and Yoder, E. A. (1988). KMS: A distributed hypermedia system for managing knowledge in organizations. *Communications of the ACM*, 31(7):820–835.
2. Anderson, K. M., Taylor, R. N., and Whitehead, Jr., E. J. (1994). Chimera: Hypertext for heterogeneous software environments. In *Proceedings of the ACM Conference on Hypertext*, pages 94–107, Edinburgh, Scotland. See also <http://www.ics.uci.edu/pub/chimera/papers/ECHT/>.
3. Berners-Lee, T. J., Cailliau, R., Groff, J.-F., and Pollermann, B. (1992). World-Wide Web: The information universe. *Electronic Networking: Research, Applications and Policy*, 2(1):52–58.
4. Carr, L., Hill, G., De Roure, D., Hall, W., and Davis, H. (1996). Open information services. *Computer Networks and ISDN Systems*, 28, pages 1027–1036.
5. Davis, H., Knight, S., and Hall, W. (1994). Light hypermedia link services: A study of third party application integration. In *Proceedings of the ACM Conference on Hypertext*, pages 41–50, Edinburgh, Scotland.
6. Grønbæk, K. and Trigg, R. (1994). Design issues for a Dexter-based hypermedia system. *Communications of the ACM*, 37(2):40–49.
7. Grønbæk, K., Bouvin, N.O., and Sloth, L. (1997). Designing Dexter-based hypermedia services for the World Wide Web. In *Proceedings of the ACM Hypertext'97 Conference*, Southampton, England.
8. Hall, W., Davis, H., and Hutchings, G. (1996). *Rethinking Hypermedia: The Microcosm Approach.* Kluwer Academic Publishers, Norwell, Massachusetts, USA.
9. Maurer, H. (1996). *Hyper-G now, HyperWave: The Next Generation Web Solution.* Addison-Wesley, Harlow, England.
10. Meyrowitz, N. (1989). The missing link: Why we're all doing hypertext wrong. In *The Society of Text: Hypertext, Hypermedia and the Social Construction of Information*, pages 107–114. MIT Press.
11. Pitkow, J. E. and Jones, R. K. (1996). Supporting the web: A distributed hyperlink database system. *Computer Networks and ISDN Systems*, 28, pages 981–992.
12. Wiil, U. K. and Demeyer, S. (1996). Proceedings of the 2nd workshop on open hypermedia systems. UCI–ICS Technical Report UCI-96-10, University of California.
13. Wiil, U. K. and Leggett, J. J. (1996). The HyperDisco approach to open hypermedia systems. In *Proceedings of the Hypertext'96 Conference*, pages 140–148, Washington D.C., USA.
14. Wiil, U. K. and Leggett, J. J. (1997). HyperDisco: collaborative authoring and Internet distribution. In *Proceedings of the ACM Hypertext'97 Conference*, Southampton, England.
15. Wiil, U. K. and Osterbye, K. (1994). Proceedings of the ECHT'94 workshop on open hypermedia systems.

Technical Report Technical Report R-94-2038, Aalborg University, Department of Computer Science.